DSC680_Project_2_Milestone_3_NoteBook_Codefile_Prashant_Raghuwan

May 7, 2022

##

Recognizing Images -Deep Learning

##

Term End Milestone-3 (Project -2)

###

CODE

###

Prashant Raghuwanshi (2223-1)

###

DSC680-T301 Applied Data Science (2223-1)

###

Professor Catie Williams

###

DSC, Bellevue University

###

05/07/2022

Each phase of the process: 1. Section **??** 1. Section **??** 1. Section **??** 2. Section **??** 3. Section **??** 4. Section 1.1.4 5. Section **??** 2. Section **??** 3. Section **??** 2. Section **??** 1. Section **??** 2. Section **??** 3. Section **??** 4. Section **??** 1. Section **??** 2. Section 5.0.1 5. Section **??** 3. Section **??** 1. Section **??** 2. Section **??** 1. Section **??** 2. Section **??** 3. Section **??** 4. Section **??** 3. Section **??** 4. Section **??** 4. Section **??** 5. Section **??** 1. Section **??** 2. Section **??** 3. Section **??** 4. Section **??** 6. Section **??** 7. Section **??**

Template Source : https://www.sv-europe.com/crisp-dm-methodology/

0.0.1 Abstract:

This term-end project-1 aims to evaluate the Students should be able to identify a business problem to address through predictive analytics. The goal is to select appropriate models and model

specifications and apply the respective methods to enhance data-driven decision-making related to the business problem. Students will identify the potential use of predictive analytics, formulate the problem, identify the right sources of data, analyze data and prescribe actions to improve not only the process of decision making but also the outcome of decisions. The current state-of-the-art Computer Vision (CV) and Machine Learning (ML) allow the detection and tracking of single objects classes (such as faces, pedestrians, or cars) in an unconstrained setting at a level that allows the realization of smart cameras that recognize smiling persons, driver assistance (pedestrian detection), surveillance applications and image-based web search. Here I am going to build an image-driven application that leverages computer vision to classify or categorize an image file based on its visual content. In essence, these applications pull valuable information and insights out of visual content.

1 1. Stage One - Determine Business Objectives and Assess the Situation

1.1 1.1 Assess the Current Situation

The use cases for image classification are virtually unlimited. Across a wide range of industries, organizations can put these AI-driven capabilities to work to streamline business processes, automate tedious manual tasks and gain insights in real-time. Please find below a few of the listed use cases for image classification : • Colleges & universities can deploy an image verification access system to keep check of unauthorized person entry into their campus. • A social media company might use video classification to categorize and annotate content. • A retailer might use image classification to enable a checkout-free store. The list of potential applications of image classification goes on and on. A commonality these applications share is deep learning, one of the key building blocks for AI solutions.

1.1.1 1.1.1. Inventory of resources

List the resources available to the project including: - Personnel: Prashant Raghuwanshi - Data: Kera Image data - Computing resources: Personal computers - Software: Jupyter Notebook(Python)

1.1.2 1.1.2. Requirements, assumptions and constraints -

Requirements : This Project is trying to make use of Deep learning techniques to automatically identify the category of a given Image. Here I am planning to feed the image to the ML model and the ML model will classify the group of images.

assumption: Here I am assuming the Preprocessing data extraction operations is not having any limitations and the images were extracted & preprocessed successfully by the image data conversion tool or Keras image preprocessing libraries.

Challenges: Major Constraints are related to used datasets and processed images, here the used datasets contain a total of 10 thousand object images, including 1 thousand for each object. However, the rapid addition of new objects in image lists would require regular re-training and deployment of the model. and this model is not trained for night vision cameras.

1.1.3 1.1.3.Risks and contingencies

- Risks include potential PII issues with respondents, which can largely be mitigated by anonymized respondent IDs. As long as a particular individual is never personally idenitified the risk of PII information leaking is mitigated.
- The process of training deep learning models is both compute- and data-intensive. Deep learning applications require massive amounts of data, fast computing, and equally fast storage, along with a lot of memory and high bandwidth networking. This is the fuel that propels AI applications forward

1.1.4 1.1.4.Terminology

CRISP-DM The Cross-Industry Standard Process for Data-Mining – CRISP-DM is a model of a data mining process used to solve problems by experts. The model identifies the different stages in implementing a data mining project, as described bellow The model proposes the following steps: • Business Understanding – to understand the rules and business objectives of the company. • Understanding Data – to collect and describe data. • Data Preparation – to prepare data for import into the software. • Modeling – to select the modeling technique to be used. • Evaluation – Evaluate the process to see if the technique solves modeling and creating rules. • Deployment – to deploy the system and train its users

1.1.5 1.1.5.Costs and benefits

Costs: • The primary cost associated with this project is the time of the people working on it. • Computing resources for modeling • Data collection and processing computing costs Benefits: • Image recognition techniques can be used to count objects such as cars or people in images. This capability can be used in traffic management and sizing crowds. • Grocers can now use computer vision capabilities to inspect and grade the quality and freshness of meats, seafood, produce, and bakery goods. • Manufacturing company might train an image and video classification system to inspect manufactured products and components for signs of defects. • A shipping company, in turn, might train a model to recognize signs of damaged boxes on a conveyor belt that is moving finished products to outbound trucks. This could be accomplished by feeding thousands of images of undamaged boxes and damaged boxes into a deep-learning algorithm

1.2 What Questions Are We Trying To Answer?

- Targeted Parameters: • Which Deep learning Model is going to fit our use case? • Which is the best target variable for our model? • What is the accuracy of the model? • Do we got any other interesting facts from datasets? Like correlations etc • Is it possible, that the ML technique can identify the multiple objects correctly?

2 2. Stage Two - Data Understanding

The second stage of the CRISP-DM process requires you to acquire the data listed in the project resources. This initial collection includes data loading, if this is necessary for data understanding. For example, if you use a specific tool for data understanding, it makes perfect sense to load your data into this tool. If you acquire multiple data sources then you need to consider how and when you're going to integrate these.

The CIFAR-10 are labeled subsets of the 80 million tiny images dataset. It is a freely available dataset on the web and can be used for education and learning purposes. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. Link: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz

2.1 2.1 Initial Data Report

Initial data collection report - List the data sources acquired together with their locations, the methods used to acquire them and any problems encountered. Record problems you encountered and any resolutions achieved. This will help both with future replication of this project and with the execution of similar future projects.

[1]:
```
# Import Libraries Required
from keras.datasets import cifar10
from keras.utils import to_categorical
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.utils import np_utils
from tensorflow import keras
from tensorflow.keras import layers
np.random.seed(35)
```

[2]:
```
##Load the dataset from keras datasets module
```

[11]:
```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

2.2 2.2 Describe Data

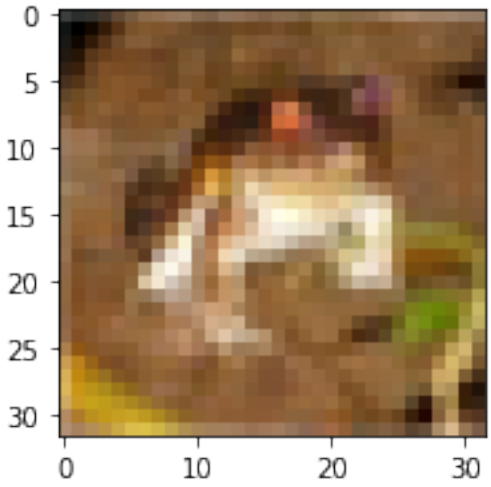
Attribute Information:

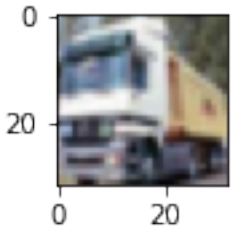
No Attributes, Image data

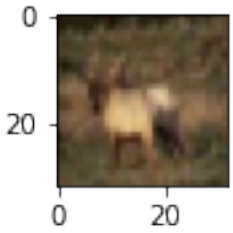
[4]:
```
#Plot some images from the dataset to visualize the dataset
```

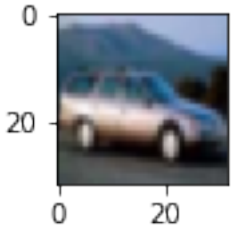
[7]:
```
n=6
plt.figure(figsize=(20,10))
for i in range(n):
   plt.subplot(330+1+i)
```

```
plt.imshow(x_train[i])
plt.show()
```









2.3 2.3 Verify Data Quality

Examine the quality of the data, addressing questions such as:

- Is the data complete (does it cover all the cases required)?
- Is it correct, or does it contain errors and, if there are errors, how common are they?
- Are there missing values in the data? If so, how are they represented, where do they occur, and how common are they?

[8]:
```
print("X_train original shape", x_train.shape)
print("y_train original shape", y_train.shape)
print("X_test original shape", x_test.shape)
print("y_test original shape", y_test.shape)
```

```
X_train original shape (50000, 32, 32, 3)
y_train original shape (50000, 1)
```

```
X_test original shape (10000, 32, 32, 3)
y_test original shape (10000, 1)
```

2.3.1 2.3.1. Missing Data

In addition to incorrect datatypes, another common problem when dealing with real-world data is missing values. These can arise for many reasons and have to be either filled in or removed before we train a machine learning model. First, let's get a sense of how many missing values are in each column

While we always want to be careful about removing information, if a column has a high percentage of missing values, then it probably will not be useful to our model. The threshold for removing columns should depend on the problem

3 3. Stage Three - Data Preperation

This is the stage of the project where you decide on the data that you're going to use for analysis. The criteria you might use to make this decision include the relevance of the data to your data mining goals, the quality of the data, and also technical constraints such as limits on data volume or data types. Note that data selection covers selection of attributes (columns) as well as selection of records (rows) in a table.

4 3.1 Label Encoding

[12]:
```
# preprocessing
x_train = x_train.astype('float32') # for division
x_test = x_test.astype('float32')
x_train /= 255 # normalise
x_test /= 255


num_classes = 10
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)
```

- All fields are already label encoded. No need to change data types.
- May potentially update to One Hot Encoding.

4.0.1 3.2.2 Drop Unnecessary Columns

Sometimes we may not need certain columns. We can drop to keep only relevent data

4.0.2 3.2 Dealing With Zeros

Replacing all the zeros from cols. **Note** You may not want to do this - add / remove as required

- Zero values were previously addressed using mean value imputation.

4.1 3.3 Construct Required Data

This task includes constructive data preparation operations such as the production of derived attributes or entire new records, or transformed values for existing attributes.

Derived attributes - These are new attributes that are constructed from one or more existing attributes in the same record, for example you might use the variables of length and width to calculate a new variable of area.

Generated records - Here you describe the creation of any completely new records. For example you might need to create records for customers who made no purchase during the past year. There was no reason to have such records in the raw data, but for modelling purposes it might make sense to explicitly represent the fact that particular customers made zero purchases.

- Do not have derivative records. No construction required.

5 4. Stage Four - Exploratory Data Analysis

5.0.1 4.1. Outliers

At this point, we may also want to remove outliers. These can be due to typos in data entry, mistakes in units, or they could be legitimate but extreme values. For this project, we will remove anomalies based on the definition of extreme outliers:

https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

- Below the first quartile $-$ 3 $*$ interquartile range
- Above the third quartile + 3 $*$ interquartile range

5.1 4.2 Initial Data Exploration

During this stage you'll address data mining questions using querying, data visualization and reporting techniques. These may include:

- **Distribution** of key attributes (for example, the target attribute of a prediction task)
- **Relationships** between pairs or small numbers of attributes
- Results of **simple aggregations**
- **Properties** of significant sub-populations
- **Simple** statistical analyses

These analyses may directly address your data mining goals. They may also contribute to or refine the data description and quality reports, and feed into the transformation and other data preparation steps needed for further analysis.

- **Data exploration report** - Describe results of your data exploration, including first findings or initial hypothesis and their impact on the remainder of the project. If appropriate you could include graphs and plots here to indicate data characteristics that suggest further examination of interesting data subsets.

5.1.1 4.2.1 Distributions

5.1.2 4.2.2 Correlations

Can we derive any correlation from this data-set. Pairplot chart gives us correlations, distributions and regression path Correlogram are awesome for exploratory analysis. It allows to quickly observe the relationship between every variable of your matrix. It is easy to do it with seaborn: just call the pairplot function

Pairplot Documentation cab be found here: https://seaborn.pydata.org/generated/seaborn.pairplot.html

5.2 4.3 Data Quality Report

List the results of the data quality verification. If quality problems exist, suggest possible solutions. Solutions to data quality problems generally depend heavily on both data and business knowledge.

- Primary data quality issue is missing values in certain parts of the data. To correct for this, we imputed the mean value of the columns into those missing fields in order to have a more complete approximation of the missing data.

6 5. Stage Four - Modelling

As the first step in modelling, you'll select the actual modelling technique that you'll be using. Although you may have already selected a tool during the business understanding phase, at this stage you'll be selecting the specific modelling technique e.g. decision-tree building with C5.0, or neural network generation with back propagation. If multiple techniques are applied, perform this task separately for each technique.

6.1 5.1. Modelling technique

Document the actual modelling technique that is to be used.

Import Models below:

[ ]:
```
#Import the required layers and modules to create our convolution neural net␣
↪architecture
```

[ ]:
```
#Create the sequential model and add the layers
```

[13]:
```
from keras import models
from keras import layers
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:
↪]))
model.add(layers.Conv2D(32, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(layers.Dropout(0.25))

model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 128)         73856
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 1, 1, 128)         147584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
dropout_6 (Dropout)          (None, 1, 1, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
```

```
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 322,602
Trainable params: 322,602
Non-trainable params: 0
_________________________________________________________________
```

6.2 5.3. Build Model

Run the modelling tool on the prepared dataset to create one or more models.

Parameter settings - With any modelling tool there are often a large number of parameters that can be adjusted. List the parameters and their chosen values, along with the rationale for the choice of parameter settings.

Models - These are the actual models produced by the modelling tool, not a report on the models.

Model descriptions - Describe the resulting models, report on the interpretation of the models and document any difficulties encountered with their meanings.

[16]:
```
#Configure the optimizer and compile the model
```

[17]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam',␣
↪metrics=['accuracy'])
```

6.3 5.2. Modelling assumptions

Many modelling techniques make specific assumptions about the data, for example that all attributes have uniform distributions, no missing values allowed, class attribute must be symbolic etc. Record any assumptions made.

-
-

[18]:
```
#View the model summary for better understanding of model architecture
```

[19]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
```

```
dropout_4 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 128)         73856
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 1, 1, 128)         147584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
dropout_6 (Dropout)          (None, 1, 1, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout_7 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 322,602
Trainable params: 322,602
Non-trainable params: 0
_________________________________________________________________
```

[20]:
```
#Train the model
```

[21]:
```
history = model.fit(x_train, y_train, epochs=10, batch_size=128,
            validation_data=(x_val, y_val),verbose=0)
```

6.4 5.4. Assess Model

Interpret the models according to your domain knowledge, your data mining success criteria and your desired test design. Judge the success of the application of modelling and discovery techniques technically, then contact business analysts and domain experts later in order to discuss the data mining results in the business context. This task only considers models, whereas the evaluation phase also takes into account all other results that were produced in the course of the project.

At this stage you should rank the models and assess them according to the evaluation criteria. You should take the business objectives and business success criteria into account as far as you can here. In most data mining projects a single technique is applied more than once and data mining results are generated with several different techniques.

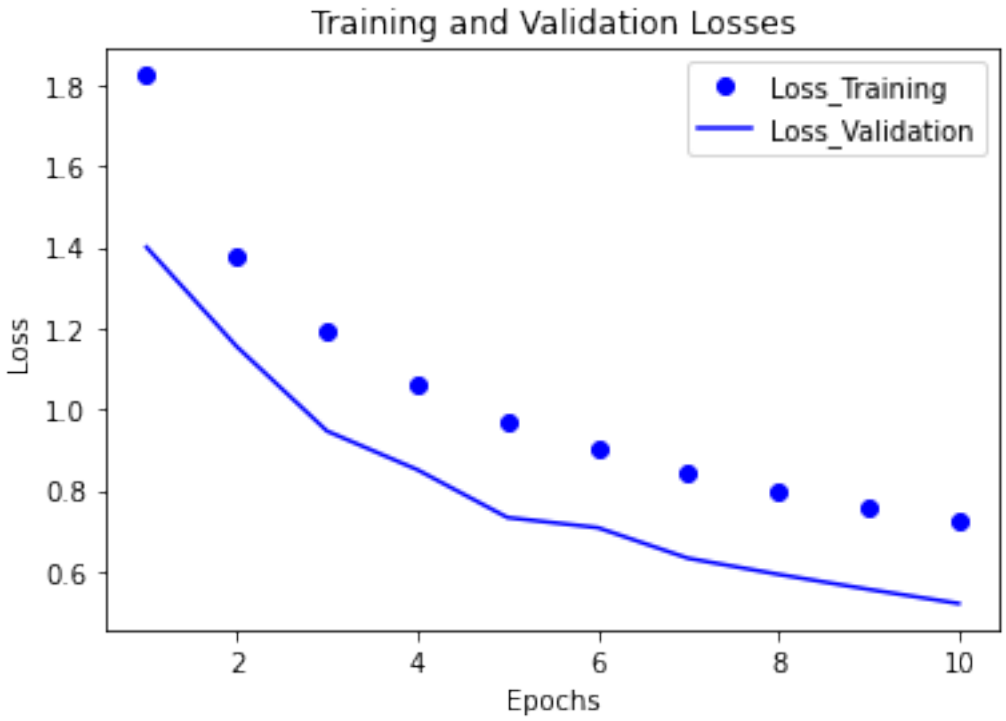
Model assessment - Summarise the results of this task, list the qualities of your generated models (e.g.in terms of accuracy) and rank their quality in relation to each other.

Revised parameter settings - According to the model assessment, revise parameter settings and tune them for the next modelling run. Iterate model building and assessment until you strongly believe that you have found the best model(s). Document all such revisions and assessments.

[22]:
```
train_loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(history.history['loss']) + 1)
```

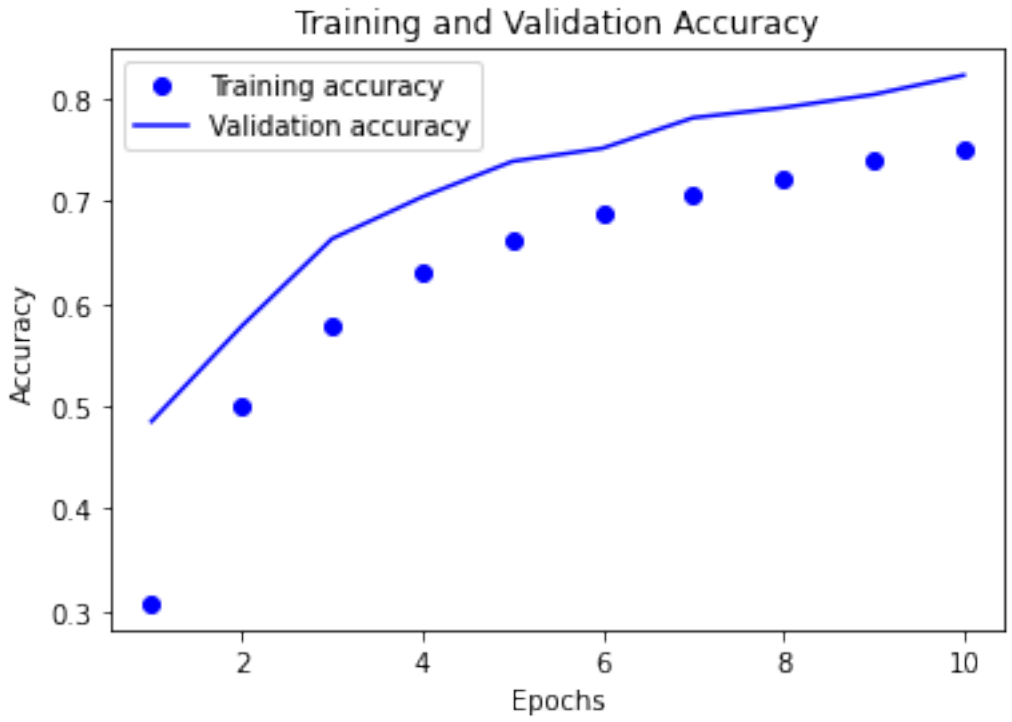
[23]:
```
plt.plot(epochs, train_loss, 'bo', label='Loss_Training')
plt.plot(epochs, val_loss, 'b', label='Loss_Validation')
plt.title('Training and Validation Losses')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



[24]:
```
train_loss = history.history['accuracy']
val_loss = history.history['val_accuracy']

epochs = range(1, len(history.history['accuracy']) + 1)
```

[25]:
```
plt.plot(epochs, train_loss, 'bo', label='Training accuracy')
plt.plot(epochs, val_loss, 'b', label='Validation accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



[26]:
```
score = model.evaluate(x_test, y_test)
print('Test accuracy: ', score[1])
```

```
313/313 [==============================] - 3s 9ms/step - loss: 0.7109 -
accuracy: 0.7548
Test accuracy:  0.754800021648407
```

The training accuracy increases linearly over time, until it reaches nearly 100%, whereas the validation accuracy stalls at 70–72%. The validation loss reaches its minimum after only ten epochs and then stalls, whereas the training loss keeps decreasing linearly until it reaches nearly 0

After fine-tuning a network, The validation accuracy curve looks much cleaner. You're seeing a nice 1% absolute improvement in accuracy, from about 96% to above 97%.

Conclusion: The generated fine tunned CNN model is the best fit for object predictions. We can use this classification model to develop Automatic object identification systems that can be designed for many processes such as university security systems.

Assumption: Here I am assuming the Preprocessing data extraction operations is not having any limitations and the images were extracted & preprocessed successfully by the image data conversion tool or Keras image preprocessing libraries. Limitations: This Model is limited to identifying the selected few varieties of objects only. To make this model more useful we need to train it with real-time images and to work with added new images this model required regular training.

Challenges: Major Constraints are related to used datasets and processed images, here the used datasets contain a total of 10 thousand object images, including 1 thousand for each object. However, the rapid addition of new objects in image lists would require regular re-training and deployment of the model. and this model is not trained for night vision cameras.

Future Uses: • Elimination of the manual comparison of images, via the introduction of an automated, quicker, and more accurate and efficient way of comparing images and UI screens. • Capability to spot minute differences that might otherwise be missed by manual inspection, including identification of pixel-level differences for fonts, colors, etc. • Solution delivery as an API service over the internet. • Does not require extensive processing capabilities or any additional expensive software or hardware. Because the AI/ML solution resides on the internet as an API service, all processing happens on the internet rather than locally.

Recommendations: Modern software can recognize a large number of everyday objects, human faces, printed and handwritten text in images, and other entities. and we'll continue witnessing how more and more businesses and organizations implement image recognition and other computer vision tasks to stand out from competitors and optimize operations. The developed image identification mode is best suited to recognize multiple individual objects and it is required to add more images in training datasets to make it more compatible with the real-time need.

[ ]: